EXHIBIT 6

Good afternoon everyone.

As you just saw, privacy is something people expect, value, and make space for every day.

I'm Mark from Privacy Engineering Team at Apple, and with my colleague, Julien, we're thrilled to tell you today about designing for privacy on our platforms and in your applications.

Privacy isn't just about closing doors.

It's also about opening doors to the trust you build with your customers.

Customers have let our devices into their homes, captured the memories in their photos, monitor their health, and transmit their thoughts and emotions, knowing that they are in control of their data.

As Tim said last year, we at Apple can and do provide the very best to our users while treating their most precious data like the precious cargo that it is.

And if we can do it, then every one of you can, and in today's talk, we'll help you design for privacy.

At Apple, privacy starts with data minimization.

Examine carefully what data do you need for the functionality of your app, and what data can you edit out? Some of the most interesting and challenging features use sensitive, personal data, and at Apple, we build these features with on device intelligence, so your device personalizes itself to you, while Apple doesn't collect this data.

When data is shared with Apple, we ask the customer.

The customer is always in control, and they always know what's being shared.

And security is foundational.

It not only protects people's data on their devices and on our servers but also enforces both the privacy protections we've built and the choices that people have made.

Some companies think that privacy is just security, that you can use or collect data so long as you protect it.

But that's short sighted, as security alone isn't adequate in the long run.

You do have a responsibility to protect the data that you collect, but it's less risky to just not collect it in the first place.

Some other companies define privacy as just transparency and control.

Let users know what you're doing with their data, and let them opt out if they don't like it.

But this is privacy after the fact, and it forces people to choose between privacy and features.

And the burden shouldn't be on individuals to exercise control to get privacy.

Because privacy is a fundamental human right.

It's essential to the functioning of good societies, and it's too important to not just work in the technologies that we use.

That's why at Apple, privacy is designed into our futures.

It's [inaudible] how they work, not something added on at the end.

At Apple, we challenge ourselves every day to build great features with privacy, and we're confident you can too.
In the rest of our talk, we'll explore privacy and the user experience.
Dive into what's new and updated in frameworks, and then we'll take a step back and show how you can overcome privacy challenges through innovation.
I'm going to talk about four user experiences today.
First, Sign In with Apple, which allows you to connect your user's experience across platforms without having to store or manage user credentials, and it's a great example of data minimization.
Apple doesn't track how people use your application because it's none of our business, and we've pared down the data that people share through Sign In with Apple to just what's needed to sign into your application.
So, ask yourself if you really need the customer's name or email address.
If you need a name for billing or shipping purposes, you can get it later, for example, through Apple Pay.
And if you don't ask for any additional data, the user has just one button to tap, and they're signed into your application.
If you do need to contact the user, for example, to provide receipts or for account recovery, we provide an easy way for you to do so.
People can be hesitant to share their real email address.
We've all seen email lists stolen or resold and then abused by spammers.
So people may give you a fake email address or one for an account they never check.
You could make the user verify their email at sign-in, but this takes them out of your application, potentially to [inaudible] their spam folder, and that's not a great experience.
With sign-in with Apple, you will get a verified email address associated with the user's AppleID.
Customers can choose to hide their email address, in which case you'll get an address managed by Apple through which we relay your emails to the customer and vice versa.
And Apple won't retain emails after we've delivered them.
For each customer, this managed address is different for each developer, so customers are in control of which developers they want to receive email from, and you're in control of who can send emails to the managed address we provide you, since you can whitelist domains or addresses that we'll accept incoming mail from.
And because both parties are in control, we think this is your best shot at getting your emails in front of your customer.
We understand you face challenges in preventing abuse and fraud from account farming.
And you may want to ask users to fill out a CAPTCHA such as this.
But these are cumbersome, and they're difficult to localize and make accessible.
Other alternatives are invasive to user's privacy, such as trying to reidentify them with personally identifiable information or fingerprinting.

With Sign In with Apple, we can leverage on-device intelligence to provide you with one bit that indicates a user is likely real.
And this flag is supported on iOS, and we provide it at account creation.
The real user status property of ASAuthorizationAppleIDCredential is how you check for this bit.
And keep in mind that a new user on a new phone will likely show up as status unknown, so you shouldn't just deny service if you see this state and provide the user and ultimate flow or a way to appeal this determination.
So that was Sign In with Apple, a great privacy friendly single sign on solution.
Now, let's talk about location.
People are protective of where they are and where they've been.
It reveals where we live, places we frequent, our patterns of life, and when those patterns change.
Like if you start attending a different set of afterparties this year with a different crowd of people.
And in iOS 13, we're making changes to help users exercise smarter control over when they're sharing their location.
Previously in iOS, there was a one-time prompt for location permissions, and at this point, the user hasn't experienced location in your app before.
So they may select don't allow and miss out on the value that location can provide in your application.
And while many of you have built experiences to explain your use of location before asking, nothing explains better than using your app.
So, in iOS 13 were introducing a new option, Allow Once, giving users the ability to try out location in your application before committing to a decision.
If the user selects Allow Once, your app will have access to location, just as if the user had selected While in Use up until the system no longer considers your app in use.
The next time the user launches your app, they'll be asked again, and if they want to give you app access without being asked, they can grant While in Use when they're comfortable doing so.
What about Always? The ability to access a user's location always is a very powerful capability, and you just may be surprised if your app asks for it right away before you've had a chance to demonstrate your app's value, the value of location in your app's value, or why you need it in the background when they're currently using your app.
Location prompts in app will now always be for just once or While in Use.
If you do request Always, we'll show the same prompt and defer asking for Always.
And if a user selects While in Use, your delegate object will be told you're authorized for always.
And iOS will continue to generate Always location events for your app, just as it did before.
Before we deliver the first Always location event to your application, we'll then prompt the user to upgrade from While in Use to Always, and you'll only get the location event if the user grants us permission.
We'll only ask users while they're on the home screen, not while they're in another application, so this may not happen immediately.

To provide increased transparency, we'll periodically remind the user that your app is accessing location in the background.
And shortly after upgrade to iOS 13, we'll remind the user for all existing apps with Always access.
The good news is, there are no capability breaking API changes, but you should test for the new Allow Once and Always behaviors to ensure that your app handles these state transitions gracefully.
And if you do use Always data, now is a good time to ensure that your use of Always provides clear user value.
We think these contextual prompts will help users make informed, smarter choices about when they're sharing their location.
Come talk to us at our lab if you have any questions, check out the video from What's New in Core Location session in the WWDC app or the Core Location lab Friday at 1 p.
m.
The third user experience I'm going to talk about is Bluetooth.
You used to only need user consent if you wanted to share information using CoreBluetooth's peripheral manager APIs in the background.
IOS 13 will now require user consent if your application uses any CoreBluetooth APIs, and this applies to apps linked against older SDKs as well.
You should provide an NSBluetoothAlwaysUsage Description purpose string in your Info.
plist to help users understand what your app needs Bluetooth for and so they can make an informed choice, and this will be required for all apps linked on or after IOS 13, or your app will crash.
Please refer to our talk last year for tips on how to write a great purpose string.
Adoption is easy.
If the user hasn't previously granted permission, then when your app instantiates a CB peripheral or central manager object, iOS will ask the user if they want to allow your app to access Bluetooth APIs.
And you can check the new authorization property on the central and peripheral manager objects for your current authorization state.
For more information, please check out what's new in CoreBluetooth Friday at 3:20.
The last user [inaudible] change is on macOS Catalina, where we're adding user consent for more types of functionality and data.
Last year in macOS Mohave, we protected these resources on the file system and required user consent to access them.
This year we're adding protection to the Desktop, Documents, and Downloads folders, iCloud Drive and Third-Party Cloud Storage as well as Network and Removable Volumes.
In addition, the camera and microphone controls introduce macOS Mohave or infusing screen recording and keyboard input monitoring in macOS Catalina as well as user control over synthetic input events and AppleEvents.
For more information, check out advances in macOS security session in the WWDC app or their lab tomorrow at 2 p.
m.

Next, I have three new frameworks with new behaviors.
First, I have some exciting news for web developers.
We introduced Intelligent Tracking Prevention, ITP, two years ago, to protect against cross-site user tracking.
On by default in Safari, it's had a big impact, and we're thrilled to see other browsers adopt similar techniques and build consensus against cross-site tracking.
We don't, however, believe that a web without free of tracking is necessarily a web without advertisements.
And last year we took a first step in building privacy friendly ad infrastructure with installed validation, which allows developers to measure conversions from advertisements to app installs.
This year, we're tackling the web.
Privacy Preserving Ad Click Attribution is a new way for advertisers and merchants on the web to measure the success of their ad campaigns without tracking individual users.
Let's say they sell running shoes on SneakerNet.
I buy ads on search.
example, and I want to know how effective those ads are at driving visits and purchases to my site.
This used to be done through cross-site tracking, which ITP now blocks.
In Privacy Preserving Ad Click Attribution, Safari connects clicks on ads with conversions on the merchant and reports these attributions to the ad provider.
Let's walk through how this happens.
Now when a user clicks on an ad on search.
example for SneakerNet, Safari will remember this for seven days.
And in that seven-day window, if the user makes a purchase on SneakerNet, the merchant or advertiser can identify this as a potential conversion, and Safari will connect this potential conversion with a previously stored ad clicks and record that conversion did occur from search.
example to SneakerNet.
Safari will then report this conversion as a delayed ephemeral post the search.
example.
Adoption is easy.
The ad provider just needs to decorate their outgoing link anchor tags with two new attributes, adDestination, the site where you expect conversions to occur.
Here, SneakerNet.
example and the adCampaignID.

Search.
example might be running many concurrent ad campaigns for SneakerNet, and the CampaignID allows you to measure them separately.
We can detect conversions without any change on the merchant site by leverage existing tracking pixels, which a search provider can then redirect the server side to a well-known URL that identifies the ConversionID as a past component.
Conversions can be anything from putting an item in a shopping cart, to creating an account, to making a purchase, and the ConversionID allows the merchant to distinguish between these different Conversion events.
To preserve the user's privacy, Safari limits the Campaign and ConversionIDs each to 64 possible values.
So, each conversion between an advertiser and merchant can only be linked to 1 of 4000 possible values.
Safari will wait 24 to 48 hours before reporting this attribution, both to aggregate multiple Clicks and Conversions and to prevent correlation with user activity on either site.
The Conversion attribution will be reported in an ephemeral session without any cookies.
And Clicks and Conversions are not recorded in private browsing.
Privacy Preserving Ad Click Attribution is available to try out in Safari Technology Preview 83.
It will be on by default in iOS 13 and macOS Catalina, and we've submitted to W3C for consideration as a web standard.
And you can see more details at webkit.
org/blog.
Next, I'm going to talk about the Speech Recognizer framework, which allows you to transcribe audio.
We've heard many requests for a way to perform transcription locally, and this is especially helpful in regulated environments such as healthcare.
In iOS 13, we're introducing offline transcription, so you can transcribe audio without internet connectivity and without sending potentially sensitive audio off device.
To adopt, you can check isAvailableForLocal Recognition on Speech Recognizer and specify requiresLocalRecognition when creating a speech recognition request.
Third, we have new ways for test users to provide feedback in TextFlight.
We understand as app developers it's really important to understand exactly where users are running into issues in your app, and we want to help you learn from your test users while preserving their privacy.
So in iOS 13, we're introducing the ability for TextFlight users to submit screen shots directly from the Screenshot UI through TextFlight.
They can also now send feedback along with crash reports so you can understand exactly what they were doing at the time of the particular crash.
Testers are in control of sharing data from their device, and this is available in TextFlight 2.
3 on iOS 13 and you can give feedback in App Store Connect.

For more information, please check out What's New in App Store Connect in the WWDC app.
Now, I'm going to turn it over to my colleague, Julien, to talk about privacy updates.
Thank you, Mark.
In this next section, let's talk about privacy updates to existing APIs.
We go through a series of five updates in rapid fashion.
First, CNCopyCurrentNetworkInfo.
The CNCopyCurrentNetworkInfo API returns the name and Mac address of the Wi-Fi access point a device is currently connected to.
This affects privacy as a Wi-Fi access point reveals a user's location.
Starting in iOS 13, CNCopy will only work for certain categories of apps, apps that declared the CNCopy entitlement and apps that configured a virtual private network.
Or apps that configured a Wi-Fi network using the NEHotspotConfiguration API or apps that obtain location permission from the user.
What this means for you is that you may not be able to access CNCopy anymore.
You can find out more details about this in the networking sessions listed below.
Second, contacts access.
Starting in iOS 13, apps that have been authorized by a user to access contacts will no longer have access to the notes field of a contact.
The notes field of a contact potentially includes sensitive details such as sneaky comments about your boss.
Most apps do not need access to this data.
If you believe you need access to it, please file an entitlement request at the following URL.
Third, Custom URL Schemes.
Previously, if your app was launched by a Custom URL Scheme, you could identify the app that launched you via an option of open URL.
This revealed applications installed on a user's device.
Starting in iOS 13, the source application is only revealed to apps from the same developer.
Fourth, Game Center.
In order to improve player privacy, Game Center introduces two new APIs.
The teamPlayerID for a team-scoped player identifier and the gamePlayerID for a game-scoped player identifier.
The existing playerID identifier is being deprecated.
This changes which identifier you may use in your gaming app.
For more information, please check out the Game Center video in the WWDC app.
And five, contextual information.
IOS doesn't know when your app is most helpful to users.

You can provide contextual information, entry point into your app via two main APIs, NSUserActivity, to promote your app in Search results, and SiriKit INInteraction to promote your app on on-device intelligent scenarios such as Maps Shortcuts and new this year in the Share Sheet.

If you do promote your app, be a good sport.

Delete content that was removed from your app.

You don't want Siri Shortcuts to make a suggestion to content that does not exist anymore.

Simply call the corresponding deletion APIs.

We just went through a series a five privacy updates to existing APIs.

In our next section, we take a step back.

We discuss how a privacy challenge can be a chance to innovate.

Privacy is fundamental to the design of every product.

At Apple, in the Privacy Team, we work with engineers all across Apple to build privacy into our products.

Once in a while, we hit a very hard problem, a feature that we really want to ship, but it's not obvious how to protect privacy.

So when that happens, we decide to challenge ourselves.

We aim to build a great feature without giving away user information, so we lean on our privacy principles to find solutions and innovate.

Today, we want to inspire you to do the same.

Let's look at three critical areas of the user experience, Machine Learning, Find My, and Home.

Let's start with Machine Learning.

There are many ways to build Machine Learning applications.

Usually we need to train a model and then evaluate it.

So the first privacy challenge with Machine Learning is how to train a machine learning model but without collecting sensitive user data while personalizing the model to every user and significantly improving the user experience.

And again, it usually starts with data, and it's usually a lot of data.

And it can be challenging to find data.

There are multiple options.

The first option is to collect data from users.

So to protect user privacy, it's important to apply best privacy practices such as randomizing identifiers, rounding or sampling.

But sometimes data is inherently sensitive, like the picture of user's face.

A second option is to run a user study.

We invite select users to test or feature in private and collect their data.

We did this ourselves to train Face ID.

Finally, their external datasets.
Some are publicly available for research purposes.
Others require careful collaboration with other companies and raise their own privacy challenges.
Once you have data, you can train a model.
You can, for example, use Create ML.
A train model is great for most users, but users are different, each with their own needs.
What if you needed to tailor, to personalize a model to specific users? This year, we introduce updatable Core ML models.
Send your updatable model to the device.
An updatable model takes in training data from the user and produces a new personalized model.
We do this ourselves with Face ID.
Face ID automatically adjusts to your appearance.
If you grow a beard or grow longer hair, Face ID still recognizes you.
And this all happens on the device.
Training uses resources though, and it may affect a user experience.
While there's a new Background Task API that allows you to schedule machine learning tasks in the background, we recommend scheduling a task within a week or less.
If you choose a date too far into the future, iOS may not schedule it.
You can learn more about the Background Task API in our session below.
But sometimes, we need more than one user's data.
We need data from many different users.
Today, I want to share with you a new technique that we are developing.
This is an example of how we innovate in privacy for machine learning.
Federated Learning is a new approach that is picking up steam in the machine learning community.
Today, we're introducing Private Federated Learning by combining Federated Learning with differential privacy.
The idea is simple, instead of collecting user data with Private Federated Learning, we collect model updates.
In other words, devices tell the server how a machine learning model should be updated.
Let's walk through an example.
It all starts with the server giving devices an original model.
We show it in white here.
Each device trains the model under local data and obtains a new model.
We show it here by changing the color.

They compute the difference between this new model and the original one and send these model updates back to Apple servers.
The server aggregates these model updates into a new enhanced machine learning model that we show here with a gradient color.
Note that in this process no user data is sent in plain, and we have no needs for any user or device identifiers.
The server then updates the devices back with a new model, and this cycle can go back and forth between the devices and the server.
From a privacy perspective, we need to analyze whether those model updates include sensitive data.
It's often considered that model updates are fine.
They're not plain user data.
But unfortunately, in some corner cases, a curious server could attempt to reconstruct the user data from model updates.
For example, here, if a user sent model updates relative to this castle, a curious server may be able to reconstruct the castle image from the model update.
So, to prevent this, we protect model updates using on device and on the server Separated Differential Privacy and Central Differential Privacy.
Differential Privacy adds noise to the model updates.
This ensures that a curious server cannot reconstruct the image anymore.
You can find more details about how we use Separated Differential Privacy in the paper below that we wrote.
Private Federated Learning can be used to improve on-device intelligence.
This is not a research experiment.
We are starting to use this technology in iOS 13 in a variety of use cases including QuickType keyboard, personalized Hey Siri, and more in the future.
Once you train a model, you want to use it.
This raises the second privacy challenge with machine learning.
How to evaluate a model without collecting user data in a fast way and without suffering from Internet connection delays.
The first way that comes to mind is to simply send user data to the server, evaluate the model there, and get back the inference.
But this means sending more user data to the server.
Instead, we prefer on-device intelligence.
We send a model down to the device.
The device evaluates it locally.
This has a lot of advantages.
It's less data sent to the servers.
It works even if you don't have Internet, and it's fast if you use Core ML.
Core ML is optimized to run fast on iOS devices and others.

It's easy to convert your existing model to a Core ML model, and this year, we expand the number of model architectures that are supported.

You can learn more in the Core ML session listed below.

On-device intelligence provides strong privacy advantages, and we like to communicate those advantages to our users in the form of privacy assurances.

A privacy assurance is a simple privacy sentence that we share with our users.

This is not a privacy policy with legalese.

For example, with Face ID, we say, Face ID data never leaves the device.

This is a clear and strong statement about our approach to privacy with Face ID.

It's easy for all users to understand.

All right, this was machine learning.

Let's talk about Find My next.

This year we announced a new Find My application that merges the Find My Friends and the Find My iPhone apps.

One feature that we were very excited about is the offline finding feature.

Here's a challenge that was brought to our attention.

A user forgot their MacBook at the public library.

The user realizes that, launches the new Find My app and attempts to find the MacBook, but it may not work because the MacBook is not connected to the Internet.

So the privacy challenge is support finding devices between Mac and iPhone or a Watch that is, even when it's not connected to the Internet when it's offline while protecting the privacy of the lost devices but also of anyone that would help you find those lost devices and making sure that Apple doesn't learn anything in the process.

And, of course, with limited impact on the battery life.

This was a long list of challenges, and so the idea for us was to rely on crowd sourcing.

We wanted to make it easy to detect a lost MacBook via Bluetooth and report its location to Apple servers.

But we couldn't rely on a static identifier broadcast over Bluetooth.

A static identifier could allow anyone to track any device at any time, and we couldn't do that.

Instead, the MacBook broadcasts and ephemeral public encryption key.

This key changes over time.

Any nearby phone can receive over Bluetooth this encryption key, compute their own location, and encrypt it with the encryption key.

Then, this encrypted data can be submitted to Apple servers.

Note here that since location is encrypted, Apple cannot see it.

Also, note that no information is provided about the nearby phone that helped find the lost device.
Finally, the owner can ask Apple later on for the corresponding encrypted data, decrypt it to obtain the location to its lost device.
With this proposal, we can share the following privacy assurance with our users.
Apple does not know the location of any offline devices or finders.
This is a big result.
We found a way to design a fantastic feature with industry leading privacy.
Next, let's talk about the Home.
The Home is a safe place, and important place for all of us.
It's a safe place we go to after work or after a long day at WWDC, and as the popularity of Smart homes increases, so does new privacy threats emerge.
Many customers have concerns about Smart home accessories, and nothing is more sensitive than security cameras in the home.
So the privacy challenge here is allowing users to view their security cameras from anywhere while allowing no one else to access the video feed, not even Apple, and still enabling Smart camera experiences.
So we worked with camera providers on two main things.
First, we rely on on-device intelligence to do computer vision in the home on the home resident device.
Be it a HomePod, an Apple TV, or an iPad, we detect people and objects directly in the home, not in the Cloud.
Second, we store camera recordings in an end-to-end encrypted manner.
This means only you can see the content from your camera, and we can make the following privacy assurance.
Only you can see the video from your camera.
Short and sweet.
It's time to wrap up the privacy session.
We talked about three privacy challenges and showed you how we solved them.
Throughout this presentation, we also talked about API updates and new features that we designed with privacy.
Every year, every release, with every product we challenge ourselves to build great features without giving away user information.
Because privacy is fundamental to the design of every product, we share privacy assurances with our users.
Simple sentences about our commitment to protect user privacy.
There are three things we want you to remember today.
First, among the first questions you should ask yourselves when working on a new project is how you will protect user data.
Ask what you should do, not what you can do with your app.
Earn the trust of your users by making it easy for them to know that their privacy is protected in your app.
Second, if you face a tough privacy problem, challenge yourself to find a solution.

Privacy is a chance to innovate.
Find a way to deliver your feature without giving away more user data.
And third, privacy is an opportunity to engage with your users.
Privacy is about people.
Don't let a privacy policy take that away from you.
Share your privacy assurances with your users.
If you have any more questions, please join us in the privacy lab right after this talk.
On behalf of the entire privacy team, thank you very much for joining us today.